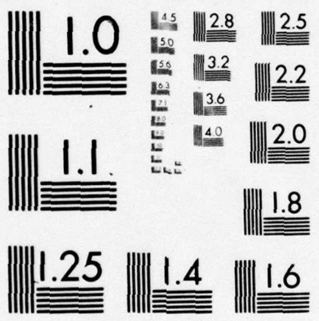
AD-A043 092 ILLINOIS UNIV AT URBANA-CHAMPAIGN CENTER FOR ADVANCED--ETC F/G 9/2
RESEARCH IN NETWORK DATA MANAGEMENT RESOURCE SHARING. OPTIMIZAT--ETC(U)
MAY 76 G G BELFORD DCA100-75-C-0021
UNCLASSIFIED CAC-197 CCTC-WAD-6504 NL

1 OF 1
ADA043092

END
DATE
FILMED
9-77
DDC


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

3

Center for Advanced Computation

CAC Document Number 197
CCTC-WAD Document Number 6504

Research in Network Data Management and Resource Sharing

Optimization Problems in Distributed Data Management

May 1, 1976

(14) CAC-197

CAC Document Number 197
(18) CCTC-WAD Document Number 6504 (19)

(6) Research in Network Data Management
Resource Sharing.

Optimization Problems in
Distributed Data Management.

by

(10) Geneva G. Belford

(9) Research rept.,

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This paper is to be published in The Proceedings of the Third International Conference on Computer Communication, Toronto, August 1976. The work was ~~performed as part of contract~~ (15) DCA100-75-C-0021 with the Command and Control Techn~~ical Center~~-WWMCCS ADP Directorate of the Defense Communications Agency.

(11) 1 May 1976 (12) 24p.

Approved for release: Peter A. Alsberg
Peter A. Alsberg, Pri~~ncipal Investi~~gator

1473

407 227 LB

TABLE OF CONTENTS

Abstract

In this paper we look at three areas of distributed data management where questions of optimization - i.e., of determining the "best" ways of doing things - seem naturally to arise. The first area, file allocation, has been the subject of considerable study, and we briefly review the literature in order to identify general techniques and trends. The second area, which we call file usage, has been studied very little. We here present two recently developed approaches for optimizing file usage in some sense. Finally, we briefly discuss the area of data organization and indicate some difficulties which must be overcome before any comprehensive attack on the problem of optimally organizing data will be possible.

Introduction

A primary purpose of this paper is to draw attention to some interesting optimization problems open for study in the new area of distributed data management. At the same time, we wish to inject a note of warning. One should not blindly undertake elaborate optimization studies just because it seems like an impressive thing to do. It is far too easy to generate complex formalisms with little utility. More study needs to go into the question of what aspects of data management are best "optimized", into the development of better mathematical models of computer systems so that they can be studied rationally, and finally, into better techniques for measurement of relevant system parameters.

Before detailing some of the problems themselves, let us describe the setting. Suppose that there are a number of computer systems, located at geographically separated sites and linked together by a communications network (such as the ARPA Network). Each of these systems possesses facilities for managing and storing data. For simplicity, we assume that there is a single very large database stored in this network of systems. Some files may be stored at several - even all - of the sites; other files may be stored at only a single site. Users at any site are able to obtain information stored at any of the other sites.

An example of such a distributed database might be the set of personnel files for a large corporation. Each plant keeps local files for its own workers. The corporation headquarters might keep a copy of the entire database. Personnel whose duties involve several plants might find their records duplicated at several sites. Now if users throughout the network may interrogate the files held at all sites, certain savings are possible. It is no longer necessary for each site to maintain complete records on all personnel that it might have some interest in. On the other hand, duplication has some positive advantages. For example, the files at one site may be lost or may become unavailable.

Once one notes that existence of the network may lead to savings, one is naturally led to optimization problems - the development of strategies to maximize the savings in some sense. The most obvious optimization problem is that of determining where to put the files in order to minimize some sort of overall cost. This problem - the so-called file allocation problem - has in fact been studied in some detail by several researchers. The next section contains a brief review of the work that has been done in this area. This review will provide some opportunity for us to note the kinds of techniques that are likely to be useful in solving other optimization problems in distributed data management. It will also allow us to point out the considerable difficulties inherent in any attempt to optimize the management of very large amounts of data. We will then proceed - in the later sections of this paper - to look at some other optimization problems which have been less heavily studied.

The File Allocation Problem

The earliest work on the network allocation problem was done by Chu [2]. Chu states the problem as follows: "Given a number of computers that process common information files, how can we allocate files so that the allocation yields minimum overall operating costs subject to the following constraints: (1) The expected time to access each file is less than a given bound, and (2) the amount of storage needed at each computer does not exceed the available storage capacity." Variables X_{ij} to describe the allocation are introduced; $X_{ij} = 1$ if the jth file is stored in the ith computer and $X_{ij} = 0$ otherwise. In order to apply constraint (1), Chu develops a reasonably comprehensive formula for access time, including queueing delays and the effect of inter-computer traffic congestion. The overall cost expression to be minimized includes costs for storage as well as for transmission. Since the variables to be determined can take on only the values zero or one, the optimal allocation may be found as the solution to a nonlinear zero-one programming problem. (In fact Chu notes that the problem may be reduced to a linear one, which may be solved by straightforward techniques.)

In a later paper [3], Chu discusses how a reliability constraint can be added to the model. The main idea is to determine in advance (from simple assumptions on failure probabilities) how many redundant copies of a file are required to achieve a desired level of reliability. This number is then inserted into the model in a simple way, and the basic scheme remains unchanged.

The difficulty with using zero-one programming to solve the file allocation problem is that it is so time-consuming as to seem impractical for very large databases. The feasibility would, of course, depend upon how often the allocation is optimized. That is, if the environment (usage patterns, network traffic) is fairly static, the optimization need only be carried out at long intervals. Another factor affecting feasibility is the number of variables X_{ij}. (This number is the product of the number of sites times the number of files.) If there are few sites, and the database is partitioned into a few large segments for allocation, then the zero-one programming problem is readily solved.

In any case, believing that zero-one programming is too costly an approach, Casey [1] developed an efficient procedure for finding a minimal-cost solution, as well as heuristic methods for finding acceptably good solutions. Casey's model differs in some respects from Chu's. Perhaps the most important difference is that Casey lets the number of copies of a file, as well as its locations, be variables. Notice that as the number of copies of a file increases, the expense of querying the file decreases, but storage and updating costs increase. Thus Casey's approach to optimization strikes a balance between these two opposing trends. A disadvantage, of course, is that the minimization may not yield enough copies for reliability.

Casey has applied his optimization algorithm to real data for the ARPA network and has thus shown the process feasible for networks of moderate size. His experiments indicated that when update traffic equals query traffic, it is most efficient to store all files at a central node. As query traffic increases relative to updates, storage at multiple nodes is indicated. These results are intuitively reasonable. Although one always expects several local minima in a complex, multivariable minimization problem, it is noteworthy that Casey's experiments reveal extremely large numbers of them (over 100 in some cases). It is clear that any optimal allocation procedure must take care to avoid being trapped in such a local minimum.

Recently, both Chu's and Casey's models have been criticized by Morgan and Levin on the basis that they do not allow for dependencies between files and programs [4,5]. That is, a program (which is itself a file) may need to make use of one or more data files. The fact that these files must interact with one another is not taken into account in the older models which assume file independence. Morgan and Levin also point out that in a heterogeneous network it may not always be possible to store a particular file at an arbitrary node. Their model takes into account this type of constraint, which also includes the possibility that the allocation may be restricted by security considerations. The algorithm used to solve the optimization problem is a systematic search procedure, along the lines suggested by Casey. Dynamic features were also introduced; that is, costs to change the file allocation were considered and balanced against savings expected from reallocation. Levin [4] proposes that expected savings may be obtained either from *a priori* knowledge of how the queries will probably change with time, or from statistical analysis of the actual queries as they occur.

In summary, the problem of optimizing a file allocation is straightforward to formulate. Solving it presents more difficulties, since the formulation is that of a large mathematical programming problem. Heuristics have been developed to obtain good, near optimal solutions. There is still a need, however, for the development of more insight into what makes a particular allocation "good." Better insight could lead to rules of thumb which a data manager can apply to carry out a good allocation without a complicated preliminary analysis.

It is highly probable that what we see here is a pattern that will reappear in other optimization problems in distributed data management. That is, the pattern of

1) straightforward problem formulation,

2) exact solution by time-consuming (but well known) techniques,

3) the development of simpler (but still time-consuming) heuristics for "good" solutions, and, finally,

4) the development of readily applicable guidelines for "acceptable" solutions.

Optimal File Usage

Once the files have been distributed throughout the network in some optimal or near-optimal fashion, the next problem is to determine how they are best used. That is, given a query or request, how is it best processed? If cost is to be minimized, the variables which might enter into such a decision include:

1) The cost C_{ij} of performing a given operation (i) at a specific site (j) which holds a copy of the relevant data. Costs may vary from site to site because of pricing policies, differences in system software or hardware, or differences in how the data are stored or indexed. For example, if a personnel file is to be searched for persons whose salary is between \$10,000 and \$15,000, the search would be much faster and less costly at a site where the file is sorted on salary or indexed on salary ranges.

2) The amount A_i of output from a given operation (i). The decision on where to carry out an operation can be affected by whether or not this output must be transported over the network. For example, suppose sites 1 and 2 both have copies of the relevant files, and the results of the request are needed at site 1. Then, even if site 2 can respond to the request (i.e., carry out the necessary operations) more cheaply than can site 1, the cost of shipping the results (which may be a

large set of records) from site 2 to site 1 may make total costs lower if the operations are performed at site 1.

Responding to a query may involve a sequence of operations (perhaps with some parallelism) to be carried out on different files. The order in which these operations are performed can have a considerable effect on total cost, but the problems which arise in trying to optimize over all possible orderings of operations are beyond the scope of this short note. Suffice it to say that this question requires further study.

Taking the order of operations as given, Willcox [7] has developed a formulation of the cost optimization problem as a zero-one programming problem. The variables Y_{ij} to be determined (and that take on only the values one or zero) simply indicate whether or not a given operation (i) is performed at a given site (j). The total cost formula consists of two terms:

1) the cost of carrying out the operations, i.e.,

   $\sum_{i,j} C_{ij}Y_{ij}$, and

2) the cost of the network traffic incurred; i.e., the sum over all operations i of the cost of transporting A_i if the operation is carried out at one site and the result is then needed at another.

If there are m operations to be carried out in fulfilling a request and n sites (on the average) at which each may be carried out, the number of variables Y_{ij} is mn. Hence there is likely to be a rather large programming problem required to optimize the handling of each query. Clearly, solving large programming problems in this setting is much more impractical than it is in file allocation, where allocations are optimized only infrequently. There is too much overhead involved even in heuristic methods for finding near-optimal solutions. Furthermore, an unrealistic amount of a priori information on database content is assumed in the presumption that one knows the parameters A_i - and even the costs C_{ij}. That is, unless there is some simple way to generate good guesses for these parameters, it would appear that optimizing retrievals in this way requires more work to get the parameters than the actual retrieval should take. Once again, one needs to look for a more simplistic basis for decision making.

Another approach to optimum file usage - and one which can be given a relatively simple mathematical formulation - is to minimize some measure of average response time by distributing the workload in an optimal manner. Let us look at a simplified version of this problem. Suppose that N sites in a network have copies of a data base and that all of these copies are up to date and equally available for use. Suppose that a query load Q is entered at just one of these sites. If we take into account network delays, does it ever improve response time to distribute the query load among the sites? Intuitively, this will occur when the query load is large enough to cause a degradation in local response that outweighs network delay.

There are two questions that immediately arise.

1) What does "load" mean?

2) How is response time affected by system "load"?

Although the first question is one which is currently the subject of considerable debate, we shall see that in the analysis carried out here a precise definition is not necessary. We may simply assume that load is a quantity which increases monotonically as jobs are put into the system.

The second question is more difficult. As a working hypothesis, we assume that response time increases linearly with system load. The justification for this assumption is the experimental and theoretical work of Scherr [6], who found that in time-sharing systems response time increases linearly with the number of users, as long as usage is fairly heavy. Thus we assume that at each site response time obeys the formula

$$\text{Response time} = a\,(\text{load} - \ell),$$

where (for simplicity) the parameters a and ℓ are taken to be the same for all sites. For this formula to be meaningful, we assume throughout the analysis that total loads are greater than ℓ.

In additon to a, ℓ, Q, and N, the following parameters are needed to analyze the question of when distribution improves response.

U = load on each computer due to updates to the database.

G_i = load on the ith computer which is not related to database use or maintenance.

T_n = increase in response time (due to network delays and overhead) when the query is sent to a remote site.

With these definitions, if the site where the queries are being generated (call this site 1) opts to respond to the entire query load itself, its total load is $U + Q + G_1$, and the single-site response time R_s is given by

$$R_s = a(U + Q + G_1 - \ell).$$

If the queries are distributed equally among the N sites, then the load on computer i is $U + Q/N + G_i$. The response time for a query answered locally is then

$$R_1 = a(U + Q/N + G_1 - \ell),$$

while the response time for a query answered at remote site i is

$$R_i = a(U + Q/N + G_i - \ell) + T_n,$$

where $i \neq 1$. The average response time $\overline{R}$ is then

$$\overline{R} = \frac{1}{N}(R_1 + R_2 + \ldots + R_N).$$

The quantity of interest is the ratio

$$R = \frac{\overline{R}}{R_s}.$$

If $R < 1$, response time is improved by distributing the queries. We therefore would like to obtain some idea of the conditions under which $R < 1$ (or, equivalently, $\overline{R} < R_s$).

For simplicity, consider the case $N = 2$. Then

$$R = \frac{R_1 + R_2}{2R_s} = 1 + \frac{G_2 - G_1 - Q + T_n/a}{2(U + Q + G_1 - \ell)}.$$

The denominator of the second term is always positive, by the assumption that total loads are larger than ℓ. Therefore the sign of the numerator determines whether R is greater than or less than one. That is, we have the result:

Distribution of the queries improves response time if and only if

$$aQ + a(G_1 - G_2) > T_n.$$

Now the parameter a is the rate of increase of response time with respect to load - the slope of the response-time curve. Thus the left side of the above inequality is just an increase in response time due to the query load and the load differential between sites 1 and 2. It is intuitively reasonable that when this quantity becomes greater than T_n (the increase in response time due to network delays and overhead), it pays to distribute. For general N the inequality becomes hardly more complex:

Distribution improves response time if and only if

$$\text{(I)}\quad aQ + a(G_1 - \overline{G}) > T_n,$$

where $\overline{G}$ is the average load at the remote sites; i.e.,

$$\overline{G} = (G_2 + G_3 + \ldots + G_N)/(N - 1).$$

An interesting point to notice is that, if the query load is sufficiently large, distributing the queries may improve response even if the local site is less heavily loaded than the remote sites.

Determination of the parameter values to use in this model poses a difficult problem. As was noted earlier, the concept of load is not well defined. Values for the G_i are difficult to come by. It may be possible, however, to make simple assumptions. For example, one could assume that all sites are approximately equally loaded. In this case, inequality (I) becomes

$$\text{(I')}\quad aQ > T_n.$$

At this point we have quantities which undoubtedly can be measured. Even though "load" is ill-defined, so that it would be hard to determine a and Q individually, the term aQ can be determined as follows. Suppose that Q increases linearly with query rate (number of queries per unit time), so that $Q = hH$, where H is the query rate and h is the load induced by unit query rate. Measure the response time $R(H_1)$ and $R(H_2)$ for two different query rates H_1 and H_2. Then, assuming that the system is sufficiently heavily loaded so that these points fall on the linear rise of the response-time curve (this point can be checked by further measurements),

$$ah \simeq \frac{R(H_1) - R(H_2)}{H_1 - H_2}.$$

Once we have a good estimate for ah, we can estimate $aHh = aQ$ for arbitrary H. Notice that this same approach will yield estimates of the left side of the inequality above even if we are not measuring a true query rate H, but only some parameter H' proportional to H. If the network is homogeneous, T_n can simply be measured by sending off some queries and comparing the response time to that for locally handled queries. A data management system can then automatically monitor query rate and response times and use inequality (I') to decide when queries should be distributed.

A simple analysis yielded definitive answers to this problem in large part because the optimization was carried out over only two choices: uniform distribution of the query load vs. processing the entire load at one site. There is perhaps a lesson to be learned here. The more the choices are restricted, the more feasible optimization becomes. More effort in the future should go into formulating simple optimization problems, with choices rationally restricted so that it is both feasible and meaningful to optimize.

It is not, of course, necessary to reduce the choices to two. The query distribution problem will be pursued somewhat further here to show that optimization questions of reasonable complexity may still have simple solutions.

Suppose that the queries, instead of being divided equally among N sites, are divided arbitrarily, a fraction w_i being handled by the ith site. Then

$$\sum_{i=1}^{N} w_i = 1,$$

and both $\overline{R}$ and $\overline{G}$ must be redefined to be weighted averages:

$$\overline{R} = \sum_{i=1}^{N} w_i R_i;$$

$$\overline{G} = \sum_{i=2}^{N} w_i G_i/(1 - w_1).$$

Inequality (I) then becomes

$$\text{(II)}\quad aQ(1 - \sum_{i=1}^{N} w_i^2)/(1 - w_1) + a(G_1 - \overline{G}) > T_n.$$

Once the concept of distributing the query load unequally among the various sites is introduced, it becomes of interest to study optimization of the distribution; that is, the determination of a set of weights $w_1, w_2, \ldots, w_N$ such that $\overline{R}$ is a minimum. Let us consider how this problem can be solved for $N = 2$. In this case $w_2 = 1 - w_1$, and we can write $\overline{R}$ in terms of the single variable w_1, the fraction of query load to be handled locally. In detail,

$$\overline{R} = aU + aQ\,(w_1^2 + (1 - w_1)^2) + w_1 G_1 a$$
$$+ (1 - w_1)G_2 a - a\ell + (1 - w_1)T_n.$$

Then

$$\frac{\partial \overline{R}}{\partial w_1} = aQ(4w_1 - 2) + a(G_1 - G_2) - T_n.$$

If we set this derivative equal to zero, we find that there is a prospective extremum at

$$w_1 = \frac{1}{2} + \frac{T_n - a(G_1 - G_2)}{4aQ},$$

$$w_2 = \frac{1}{2} - \frac{T_n - a(G_1 - G_2)}{4aQ}.$$

This point is readily shown to be a minimum, as desired. Furthermore, the weights w_1 and w_2 can be seen to be positive (as is necessary for this result to be meaningful) under a wide range of conditions; for example, if $G_1 = G_2$ and inequality (I') holds.

Some interesting conclusions can immediately be read from the equations for w_1 and w_2. First note that if the loads are equal ($G_1 = G_2$)

the local site should always handle more than half of the queries. Only when $T_n = a(G_1 - G_2)$, so that the network delay equals the increase in response time due to load differential, should the query loads be equalized. And only when T_n is less than $a(G_1 - G_2)$ should the local site send off more queries than it keeps.

It must again be emphasized that careful measurements are required for these relationships to be useful for real decision making. It is easy to estimate that T_n, aQ, and $a(G_1 - G_2)$ may all, under reasonable assumptions, be on the order of one to two seconds. This information is not at all helpful for developing long-term strategies, but merely demonstrates that the optimum decision on query sharing should be done dynamically and only after monitoring current system usage and response.

The above analysis for optimum distribution strategy was done for the N = 2 case. The general case can be handled similarly, but is more complicated because of the multi-variable minimization. Setting the derivatives to zero and solving yields the following equations for $i \neq 1$.

$$w_i = \frac{1}{2}(1 - \sum_{j \neq i,1} w_j) - \frac{T_n - a(G_1 - G_i)}{4aQ}$$

Clearly this reduces to the simple formula found above for N = 2, i = 2. But in this case we have a set of simultaneous, linear, algebraic equations in $w_2, \ldots, w_N$ to solve. It is a simple matter to show that this set of equations has a unique solution, readily obtainable by computation, and that this solution does minimize $\overline{R}$. Again, it is necessary to check that the weights w_i that are computed are all positive, in order that the solution be meaningful.

All of the analysis above has been under the assumption that the query load all originates at a single site. Suppose instead that each site i generates some fraction f_i of the total query load Q. Site i then distributes its query load with a strategy described by weights $w(i)_1$, $w(i)_2, \ldots, w(i)_N$. The query load that a site i must respond to is then given by

$$Q_i = \sum_j f_j Q w(j)_i,$$

so that site i's response time (i.e., time to respond to a query) is

$$R_i = a(U + Q_i + G_i - \ell).$$

From the point of view of site j, the average response time seen is computed as

$$\overline{R}_j = \sum_i w(j)_i R_i + (1 - w(j)_j) T_n,$$

since a network delay of T_n is observed for the fraction of queries answered remotely. Now to get an average response time for queries originated throughout the network, we must take another weighted average:

$$\overline{R} = \sum_j f_j \overline{R}_j.$$

Combining the preceding four equations, we get an equation for $\overline{R}$ in terms of the N^2 variables $w(j)_i$. As above, we can carry out an optimization analysis or compare various strategies. (For example, the strategy where each site handles its own queries is described by $w(j)_i = 1$ when $i = j$ and $w(j)_i = 0$ otherwise.) We will not go into further details on this generalization in this brief report.

Optimization of Data Organization

We have looked at two optimization problems for distributed databases - the file allocation problem and the file usage problem.

Doubtless many others can be suggested. For example, one important problem which must be faced by data managers is that of deciding on data organization and structure, including indexing. The possible choices are expanded many-fold in a distributed environment, where different sites may have different indices, etc. (The file allocation problem can be thought of as just one small part of the comprehensive data organization problem.) Attempts to automate or optimize decisions on data organization have so far been carried out only in very restricted contexts - single systems, small sets of data, very limited choice of alternatives, etc. One would like to think that someday automatic organization of large distributed databases will be commonplace. But the problem of automating decisions on this large a scale is enormous.

Consider just one small subproblem. What should be the underlying basis for decisions on data organization? There are two approaches. One is to organize on the basis of database content. This approach is taken, for example, in document retrieval, where documents with similar content (as determined by keywords) are clustered together. The other approach is to organize on the basis of observed usage patterns. This sounds very attractive; only by knowing precisely how the database will be used can one begin to optimize its structure from the point of view of the users. Although research along these lines is beginning, there remain many problems. For example, how should one define the concept of "usage pattern" so that it is both conveniently measured and relevant to the decision on data organization? This question will be an important one in studying, for example, the problem of optimal indexing. Clearly, the more elaborate the indexing, the faster retrieval should be. On the other hand, updates will become more expensive and time consuming. Thus usage patterns for this application must include comprehensive information on updates as well as retrievals.

Finally, the example of indexing points up another consideration which must be kept in mind by anyone asking for the "best" way to solve some data management problem. Unless there are well defined tradeoffs - such as that between update and retrieval cost - or well formulated constraints, there may be no optimum, but only a continual improvement as money and resources are poured into solving the problem.

Summary and Conclusions

In this paper we have looked at three areas of distributed data management where questions of optimization - i.e., of determining the "best" ways of doing things - seem naturally to arise. The first area was that of file allocation, or determining the best sites in the network for the data. Although this problem has been studied for a number of years, and, given enough time and information, one can obtain an optimum by straightforward techniques, there still is room for the development of readily applicable guidelines that data managers can use to obtain "good" practical solutions.

Next, we looked at the problem of optimizing data base usage. Once several copies of the data - or portions thereof - exist in the network, we would like to determine the best strategy for answering a query. Hardly anything exists in the literature on this problem. Two new approaches were considered here - minimization of "cost" and of response time. The former leads to a formulation very analogous to that for file allocation, but for which the parameters are even harder to obtain. The latter seems to lead to something closer to a readily applicable rule of thumb, but at the expense of being perhaps too simplistic.

Third, we looked briefly at the problem of optimizing data organization - indexing, structure, etc.. This is a problem of importance for a single site database as well as for a distributed one. In spite of considerable study of certain limited aspects of the problem, there is not really enough accomplished to convince one that optimal organization of databases will ever be a reality. Before this goal becomes feasible, much more preliminary work needs to be done - on problems such as developing usable, rigorous definitions for data structures and for usage patterns. In fact, we have not even reached the point where we can properly define what we mean by optimization of data organization. Persons responsible for data management should give careful thought to identifying those subproblems for which the development of good, though perhaps suboptimal, solutions will be most valuable. In the short term, we can expect noticeable progress and practical payoffs only on such subproblems. Undoubtedly, some of these subproblems will prove to have been studied in some context in the past, but others will require the breaking of fresh ground.

References

1. Casey, R.G., "Allocation of Copies of a File in an Information Network", AFIPS Conference Proceedings 40, 1972, pp. 617-625.

2. Chu, W.W., "File Allocation in a Multiple Computer System", IEEE Trans. Computers C-18 (10), 1969, pp. 885-889.

3. Chu, W.W., "Optimal File Allocation in a Computer Network", in Computer-Communications Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, Englewood Cliffs, N.J., 1973, pp. 82-84.

4. Levin, K.D., "Organizing Distributed Data Bases in Computer Networks," Ph.D. Dissertation, University of Pennsylvania, 1974.

5. Morgan, H.L. and Levin, K.D., "Optimal Program and Data Locations in Computer Networks", Report 74-10-01, Dept. of Decision Sciences, The Wharton School, University of Pennsylvania, 1974.

6. Scherr, A.L., An Analysis of Time-shared Computer Systems, MIT Press, 1967.

7. Willcox, D., "Optimal Query Strategies", Internal Technical Memo, Center for Advanced Computation, University of Illinois at Urbana-Champaign, 1975.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CAC Document Number 197 CCTC-WAD Document Number 6504 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Optimization Problems in Distributed Data Management		5. TYPE OF REPORT & PERIOD COVERED Research Report
		6. PERFORMING ORG. REPORT NUMBER CAC#197 ✓
7. AUTHOR(s) Geneva G. Belford		8. CONTRACT OR GRANT NUMBER(s) DCA100-75-C-0021 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Advanced Computation ✓ University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Command and Control Technical Center WWMCCS ADP Directorate 11440 Isaac Newton Sq., N., Reston, VA. 22090		12. REPORT DATE May 1, 1976
		13. NUMBER OF PAGES 22
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Copies may be obtained from the
National Technical Information Service
Springfield, Virginia 22151

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
No restriction on distribution

18. SUPPLEMENTARY NOTES
None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
distributed data management
file allocation
optimal file usage

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report briefly reviews three areas of distributed data management where optimization questions naturally arise. The areas, which are in widely different stages of development, are file allocation, file usage, and data organization.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)